09/769,953

L Number	Hits	Search Text	DB	Time stamp
1	12773	((input/output) or i/o) adj (bus or interconnect)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:32
3	85033	(detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:39
4	1	(((input/output) or i/o) adj (bus or interconnect)) same ((first or primary or master)adj bus adj portion) same ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:40
5	6547	parity adj error$	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:41
6	2764	(terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:52
7	6	(((input/output) or i/o) adj (bus or interconnect)) same (parity adj error$) same ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:52
8	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:01
9	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:01
10	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:01
11	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:01

12	3	((first or primary or master)adj bus adj portion) and (parity adj error$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:06
13	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:06
14	1	((first or primary or master)adj bus adj portion) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:06
15	8	(((input/output) or i/o) adj (bus or interconnect)) and ((first or primary or master)adj bus adj portion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:07
2	33	(first or primary or master)adj bus adj portion	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:28
16	3	(low or high) adj bus adj parity	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:36
17	0	initiator adj3 interfce	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:36
18	203	initiator adj3 interface	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:38
19	47953	device adj3 interface	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:38
20	1026	acknowledge with transaction	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:40

21	14	(((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:50
22	1	((((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)) and ((first or primary or master)adj bus adj portion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:50
23	1	((((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)) and (initiator adj3 interface)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:50
24	3	((((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)) and (device adj3 interface) and (acknowledge with transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:52
25	3	(((((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)) and (device adj3 interface) and (acknowledge with transaction)) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:52
26	4005	(714/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:53
27	3093	(710/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:53
28	677	(712/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:53

29	7694	((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:54
30	6	(device adj3 interface) same (acknowledge with transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:54
31	109	(((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:54
32	2	(((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)) and ((device adj3 interface) same (acknowledge with transaction))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:55
33	4	(((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)) and ((first or primary or master)adj bus adj portion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:55
34	1	(((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)) and ((first or primary or master)adj bus adj portion) and (device adj3 interface) and (acknowledge with transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:56
35	1	((device adj3 interface) same (acknowledge with transaction)) and ((first or primary or master)adj bus adj portion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:56
36	1	(initiator adj3 interface) and ((first or primary or master)adj bus adj portion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:57
37	1	(device adj3 interface) and (acknowledge with transaction) and ((first or primary or master)adj bus adj portion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:57
38	20	(device adj3 interface) and (acknowledge with transaction) and (((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:57

39	1	((device adj3 interface) and (acknowledge with transaction) and (((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.))) and ((((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:57

09/769,953

L Number	Hits	Search Text	DB	Time stamp
-	12	(("5857086") or ("5867645") or ("5889970") or ("5892964") or ("5923860") or ("6018810")).PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 16:43
-	14	(("5361267") or ("5701409") or ("5724528") or ("5764924") or ("5781918") or ("5867645") or ("5884027")).PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:09
-	3983	(accelerat$4 adj graphic$ adj port) or AGP	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:11
-	346129	(peripheral adj component ad interconnect) or PCI	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:12
-	1976	((accelerat$4 adj graphic$ adj port) or AGP) with ((peripheral adj component ad interconnect) or PCI)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:12
-	3835044	detect$ or monitor$4 or track$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:13
-	5084838	error$ or fault$4 or problem$ or fail$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:14
-	174525	(detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:15
-	9999	(first or primary or master) adj bus	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:16
-	46	((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:16

-	0	(((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) with (((accelerat$4 adj graphic$ adj port) or AGP) with ((peripheral adj component ad interconnect) or PCI))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:17
-	1	(((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) and (((accelerat$4 adj graphic$ adj port) or AGP) with ((peripheral adj component ad interconnect) or PCI))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:17
-	1	(((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) and ((accelerat$4 adj graphic$ adj port) or AGP) and ((peripheral adj component ad interconnect) or PCI)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:18
-	4005	(714/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:18
-	3093	(710/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:18
-	677	(712/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:19
-	7694	((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:19
-	12	(((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) and (((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:31
-	6547	parity adj error$	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:31
-	15	(((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) and (parity adj error$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:31

-	12533	(input/output or i/o)adj bus	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:32
-	4	(((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) and ((input/output or i/o)adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:35
-	1	((((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) and (parity adj error$)) and ((input/output or i/o)adj bus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:35
-	10	((((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) and (parity adj error$)) not (((((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) and (((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.))) or ((((detect$ or monitor$4 or track$4) adj3 (error$ or fault$4 or problem$ or fail$4)) with ((first or primary or master) adj bus)) and ((input/output or i/o)adj bus)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/02 17:35

09/769,953

L Number	Hits	Search Text	DB	Time stamp
1	12773	((input/output) or i/o) adj (bus or interconnect)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:32
3	85033	(detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:39
4	1	(((input/output) or i/o) adj (bus or interconnect)) same ((first or primary or master)adj bus adj portion) same ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:40
5	6547	parity adj error$	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:41
6	2764	(terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:52
7	6	(((input/output) or i/o) adj (bus or interconnect)) same (parity adj error$) same ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 09:52
8	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:01
9	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:01
10	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:01
11	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:01

12	3	((first or primary or master)adj bus adj portion) and (parity adj error$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:06
13	1	((first or primary or master)adj bus adj portion) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:06
14	1	((first or primary or master)adj bus adj portion) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:06
15	8	(((input/output) or i/o) adj (bus or interconnect)) and ((first or primary or master)adj bus adj portion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:07
2	33	(first or primary or master)adj bus adj portion	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:28
16	3	(low or high) adj bus adj parity	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:36
17	0	initiator adj3 interfce	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:36
18	203	initiator adj3 interface	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:38
19	47953	device adj3 interface	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:38
20	1026	acknowledge with transaction	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:40

21	14	(((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:50
22	1	((((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)) and ((first or primary or master)adj bus adj portion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:50
23	1	((((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)) and (initiator adj3 interface)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:50
24	3	((((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)) and (device adj3 interface) and (acknowledge with transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:52
25	3	(((((input/output) or i/o) adj (bus or interconnect)) and ((detect$4 or track$4 or monitor$) adj (error$4 or fault$4 or problem$)) and (parity adj error$) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)) and (device adj3 interface) and (acknowledge with transaction)) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:52
26	4005	(714/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:53
27	3093	(710/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:53
28	677	(712/?).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:53

29	7694	((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:54
30	6	(device adj3 interface) same (acknowledge with transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:54
31	109	(((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)) and ((terminat$4 or halt$4 or interrupt$4 or stop$4) adj3 transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:54
32	2	(((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)) and ((device adj3 interface) same (acknowledge with transaction))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:55
33	4	(((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)) and ((first or primary or master)adj bus adj portion)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:55
34	1	(((714/?).ccls.) or ((710/?).ccls.) or ((712/?).ccls.)) and ((first or primary or master)adj bus adj portion) and (device adj3 interface) and (acknowledge with transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/03 10:55

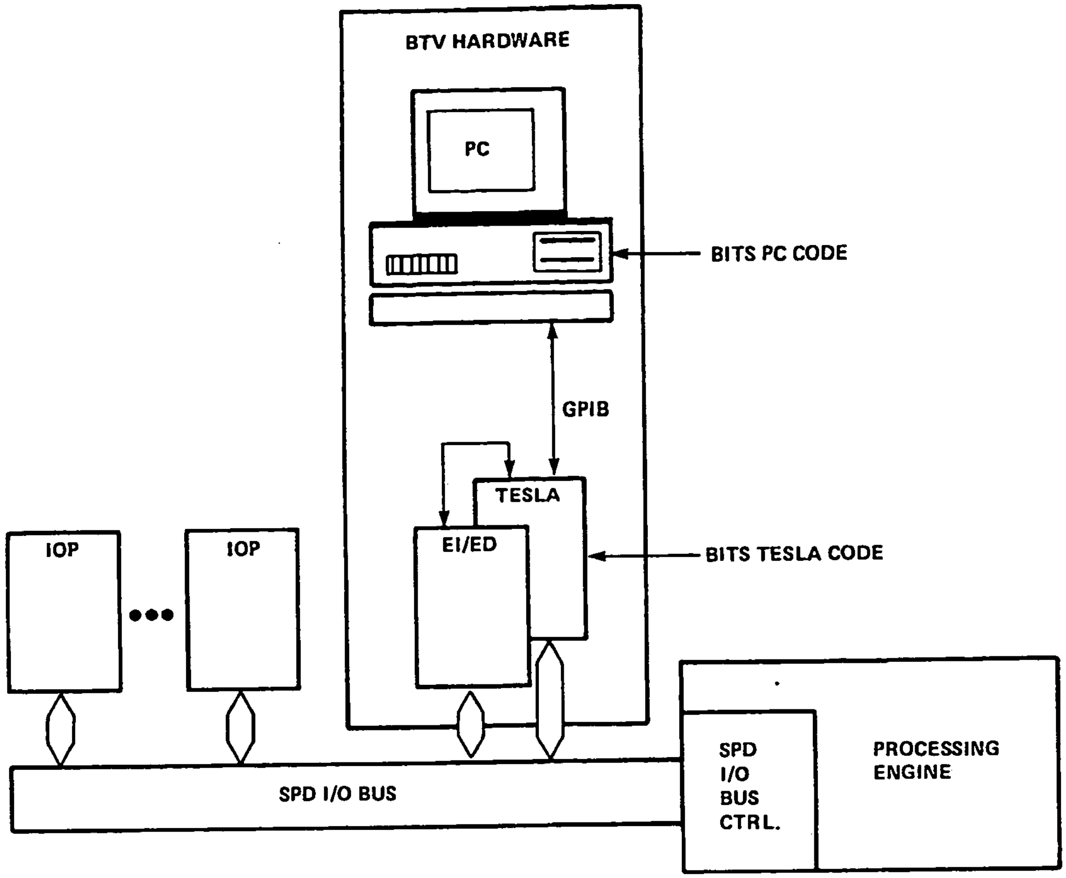

FIG. 1 BITS HARDWARE STRUCTURE

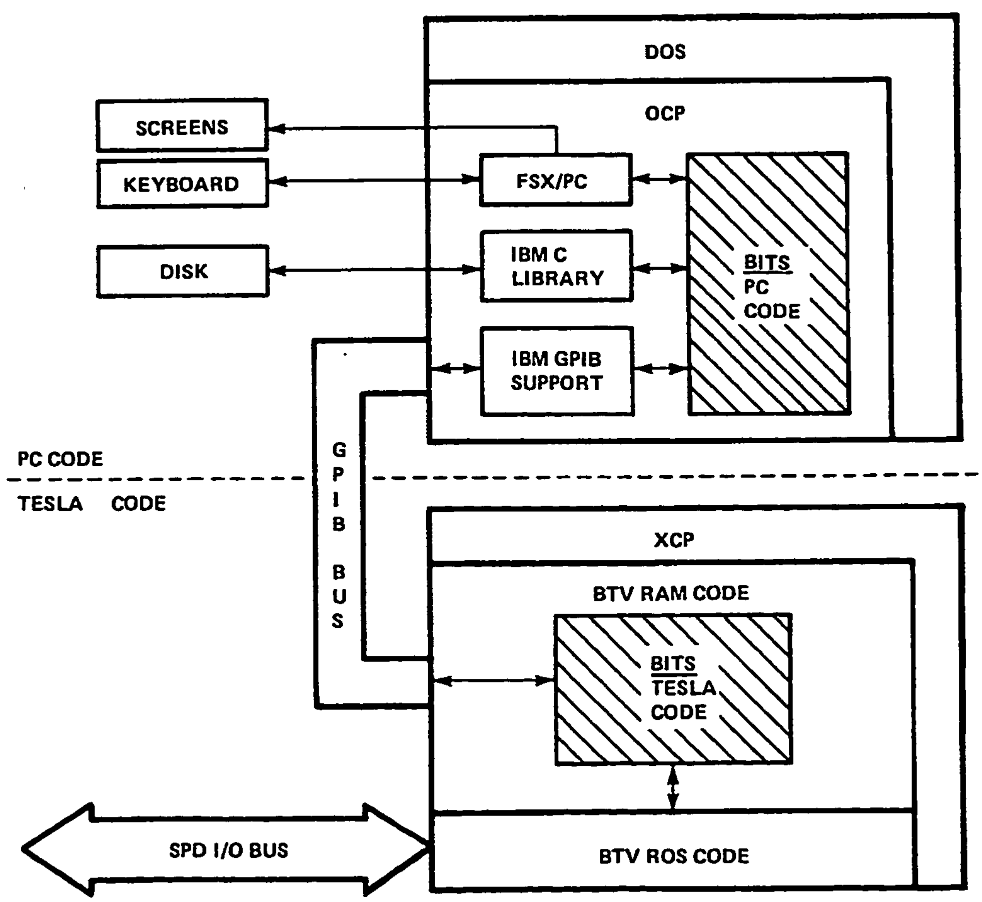

FIG. 2 BITS SOFTWARE STRUCTURE

US-PAT-NO: 4852048

DOCUMENT-IDENTIFIER: US 4852048 A

TITLE: Single instruction multiple data (SIMD) cellular array processing apparatus employing a common bus where a first number of bits manifest a **first bus portion** and a second number of bits manifest a second bus portion

---------- KWIC ---------

TITLE - TI (1):
Single instruction multiple data (SIMD) cellular array processing apparatus employing a common bus where a first number of bits manifest a **first bus portion** and a second number of bits manifest a second bus portion

Detailed Description Text - DETX (12):
The test interface and PROM 244 conveys **parity error** information off chip during normal chip operations, and conveys manufacturing test data during system initialization.

Detailed Description Text - DETX (54):
The PROM 1009 is shifted out by clock phase B when reset drops while CS is high. See FIG. 7. The **parity error** bus is output when the reset is low and there is no shifting at which time the buffer 1012 pullup transistor is disabled to provide a wired or connection. It is noted that the chip can be used even if there are a number of defects.

Detailed Description Text - DETX (101):
The local memory data is connected to the bus 3005 through transceiver 3001. Data being received from the local memory is checked for parity by the parity generator/checker 3000 ln the event a **parity error** is detected, the cell **parity error** signal is asserted, being captured by the sticky **parity error** flip flop which is shown in FIG. 23. The parity generator checker is of a conventional design as having a tree of exclusive OR gates to check for odd parity. The

interface to the high speed I/O data bus through transceiver 3008 is double
buffered. Registers IMDRA 3006 and IMDRB 3007 are under control of the I/O
controller 3009.

Detailed Description Text - DETX (119):
In addition there is a processor enable which is controlled by the vector
IF/ELSE logic. If this bit is true, then storage in the cell may be updated
during an instruction, although the storage associated with the vector IF/ELSE
logic is updated regardless of the state of this bit. The sticky **parity error**
bit is set whenever data is received by the cell that contains a **parity error,**
and this bit is the OR of itself and the **parity error** bit so that once set, it
will remain stuck until cleared by the program or loading the PSW. While any
sticky **parity error** bit is set, the **parity error** flag at the output of the chip
will be set.

Detailed Description Text - DETX (134):
Referring to FIG. 23, there is shown the status register alternate inputs.
Each of the bits of the status register ALT I/O is received by a latch. Bach
latch contains a pass transistor 4400 clocked by clock phase A and a buffer as
4401 to produce the shift register bus (SR bus) output. Tri-state buffer 4402
clocked by clock phase B and alternate load X receives the external shift
register bus and passes the data back to the status register via the I/O lines.
X varies with the bit as shown since the bits are written in several groups.
Multiplexer 4403 under control of the PLA bus select one of two inputs to be
passed back to the carry flip flop. OR gate 4404 computes the sticky overflow
bit. Or gate 4405 computes the cell sticky **parity error**. In addition, the
sticky parity bit is gated with Active by AND gate 4406 to drive pull-down
transistor 4407. This transistor drives the parity bus which is common to all
of the cells to indicate a **parity error** off chip. NOR gate 4408 receives the
four slice configuration mask bits to produce the not-active output which is
used throughout the path logic.

Claims Text - CLTX (3):
a common bus for said row capable of propagating data 2N bits wide with a first number of bits manifesting a **first bus portion** and a second number of bits manifesting a second bus portion, said first and second portions of said bus being connected to each of said processing cells in said row, wherein each of said processing cells is capable of transferring and receiving data N bits wide to and from either one of said first and second bus portions of said common bus; and

Claims Text - CLTX (4):
instruction means connected to each of said processing cells in said row for selectively providing an instruction to each process cell to select a mode in which said processor cell transfers and receives data N bits wide to and from one of said **first bus portion** or said second bus portion;

Claims Text - CLTX (5):
logic means associated with each of said processor cells in said row which is operative in response to a respective instruction provided by said instruction means to cause the respective processor cell to transfer and receive data from said common bus on either said **first bus portion** or said second bus portion according to its respective selected mode.

Claims Text - CLTX (9):
5. The cellular array according to claim 4, additionally comprising a row decoder associated with each row of said array and coupled to each of said memories of each cell to enable each cell in a row to be selected according to operation thereof with said second bus portion or said **first bus portion** of said common bus.

US-PAT-NO: 6108740

DOCUMENT-IDENTIFIER: US 6108740 A

TITLE: Method and apparatus for terminating a bus such that stub length requirements are met

---------- KWIC ---------

Brief Summary Text - BSTX (4):

The SCSI bus is a local Input/Output (**"I/O") bus** that can be operated over a wide range of data rates. The primary object of the SCSI bus interface is to provide host computers with device independence within a class of devices. Accordingly, different disk drives, tape drives, printer, optical media drives, and other peripheral devices can be added to a host computer without requiring modifications to generic system hardware or software. Standards have been defined by the American National Standards Institute (ANSI) for different types of SCSI busses. In particular, the ANSI document, SCSI Parallel Interface-2, describes in detail the SCSI-2 standard, the disclosure of which is hereby incorporated by reference.

Brief Summary Text - BSTX (14):

Pursuant to yet another embodiment of the present invention, there is provided a bus controller for interfacing buses. The bus controller includes a first bus port, a second bus port, a first bus channel, a control circuit, a first terminating circuit, and a second terminating circuit. The first bus port and the second bus port are both operable to couple to devices. The first bus channel is coupled between the first bus port and the second bus and includes a **first bus portion** of a first bus. The first terminating circuit is operable to selectively terminate the first bus in response to a first control signal and is coupled to a first stub connection of the first bus channel via a first group of conductor lines that are no greater than a first maximum stub length allowed for the first bus. The first stub connection is located

on the
first bus channel within the first maximum stub length from the first bus port.
The second terminating circuit is operable to selectively terminate the first
bus in response to a second control signal and is coupled to a second stub
connection of the first bus channel via a second group of conductor lines that
are no greater than the first maximum stub length. The second stub connection
is located on the first bus channel within the first maximum stub length from
the second bus port. The control circuit is operable to control the first bus
and is coupled to the first bus channel with a first stub that is no greater
than the first maximum stub length.

Claims Text - CLTX (39):
a first bus channel coupled between said first bus port and said second bus,
said first bus channel comprising a **first bus portion** of a first bus;

US-PAT-NO: 5987488

DOCUMENT-IDENTIFIER: US 5987488 A
See image for Certificate of Correction

TITLE: Matrix processor

---------- KWIC ---------

Detailed Description Text - DETX (3):
The microcontroller 3 is connected to the program memory 2 to manage the
matrix processing algorithm and the different elements of the processor 1 to
perform the algorithm. The microcontroller 3 is also connected to an
input/output bus 15 and a control bus 16. The **input/output bus** 15 is connected
to the data memory 4 and to the input/output interface 5. By means of the
input/output bus 15, the microcontroller 3 controls and checks the input and
the output of the data of the data memory 4 with respect to the exterior of the
processor 1 through the data input/output interface 5 as well as with respect
to the computation units 6 to 14. The command bus 16, in the present example,
has 32 wires, of which 14 wires set up a connection of the microcontroller 3
with all the computation units 6 to 14. The other 18 wires of the control bus
correspond to nine pairs of wires setting up connections between the
microcontroller 3 and, respectively, each of the nine computation units 6 to
14.

Claims Text - CLTX (5):
a **first bus portion** having a first group of wires connected to the plurality
of computation units to convey a common instruction thereto, and

Claims Text - CLTX (23):
a **first bus portion** having a first group of wires connected to the plurality
of computation units to convey a common instruction thereto, and

Claims Text - CLTX (34):
connecting a **first bus portion** having a first group of wires to the
plurality of computation units to convey a common instruction thereto,
and

DERWENT-ACC-NO: 1997-057229

DERWENT-WEEK: 199707

TITLE: **Input/output bus** for workstation, PC - has bus bridge circuit which is provided between two bus portions to form loop

PATENT-ASSIGNEE: HITACHI LTD[HITA]

PRIORITY-DATA: 1995JP-0111525 (May 10, 1995)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
JP 08305658 A	November 22, 1996	N/A	007	G06F 013/36

APPLICATION-DATA:

PUB-NO	APPL-DESCRIPTOR	APPL-NO	APPL-DATE
JP 08305658A	N/A	1995JP-0111525	May 10, 1995

INT-CL (IPC): G06F013/36

ABSTRACTED-PUB-NO: JP 08305658A

BASIC-ABSTRACT:

The **input/output bus has a first bus portion** connected between processor bus (3) through a first processor bridge circuit.

A second bus portion is connected between the processor bus through a second processor bridge circuit. A bus bridge circuit is provided between the two bus portions to form a loop.

ADVANTAGE - Reduces distance between input/output devices. Prevents loss in bandwidth of processor bus.

CHOSEN-DRAWING: Dwg.1/5

TITLE-TERMS: INPUT OUTPUT BUS BUS BRIDGE CIRCUIT TWO BUS PORTION FORM LOOP

DERWENT-CLASS: T01

EPI-CODES: T01-J07A1;

SECONDARY-ACC-NO:
Non-CPI Secondary Accession Numbers: N1997-047103

US-PAT-NO: 6446151

DOCUMENT-IDENTIFIER: US 6446151 B1

TITLE: Programmable time slot interface bus arbiter

---------- KWIC ---------

Claims Text - CLTX (20):
20. The bus arbitrated system according to claim 19, wherein: said system
bus includes a **first bus portion** electrically isolatable from a second bus
portion; said **first bus portion** is in direct communication with at least one
of said plurality of asynchronous data ports; and said second bus portion is
in direct communication with the remainder of said plurality of asynchronous
data ports.

US-PAT-NO: 5590372

DOCUMENT-IDENTIFIER: US 5590372 A

TITLE: VME bus transferring system broadcasting modifiers to
multiple devices and the multiple devices simultaneously
receiving data synchronously to the modifiers without
acknowledging the modifiers

---------- KWIC ---------

Claims Text - CLTX (2):
a VME bus including at least a **first bus portion** and a second bus portion;

Claims Text - CLTX (3):
a first device attached to at least the **first bus portion** and the second bus portion and comprising

Claims Text - CLTX (7):
means for sequentially sending data words via the **first bus portion** synchronously with said clock pulses without receiving an acknowledgment from any recipient from a time of said first device requesting said control until completion of said broadcast; and

Claims Text - CLTX (10):
receiving means, responsive to the broadcast identifier, for receiving said data words via the **first bus portion** synchronously with said clock pulses without replying with an acknowledgment from a time of said first device requesting said control until completion of said broadcast, said second and third devices simultaneously receiving said data words.

Claims Text - CLTX (12):
the **first bus portion** is a data bus portion, and the second bus portion is an address modifier bus portion; and

Claims Text - CLTX (18):
4. A device for digital communication via a VME bus, said bus

comprising at
least a **first bus portion** and a second bus portion, the device
comprising:

Claims Text - CLTX (21):
   means for sending an indicator for a broadcast via the second bus portion to
cause a plurality of other devices attached to the bus to simultaneously
receive a sequence of data words via the **first bus portion** synchronously with
said clock pulses and not reply with an acknowledgment from a time of the first
said device requesting said control until completion of said broadcast; and

Claims Text - CLTX (22):
   means for sending said data words via the **first bus portion** synchronously
with said clock pulses without requiring an acknowledgment signal from any of
said other devices from a time of said first device requesting said control
until completion of said broadcast.

Claims Text - CLTX (24):
   the **first bus portion** is a data bus portion, and the second bus portion is
an address modifier bus portion; and

US-PAT-NO: 6557068

DOCUMENT-IDENTIFIER: US 6557068 B2

TITLE: High speed peripheral interconnect apparatus, method and system

---------- KWIC ---------

Brief Summary Text - BSTX (7):

I. Registered PCI Transaction Protocol A. Overview of Registered PCI B. Transaction Comparison Between Registered PCI and Conventional PCI C. Registered PCI Transaction Protocol 1. Transaction Sequences 2. Allowable Disconnect Boundaries (ADB) and Buffer Size 3. Wait States 4. Addressing, Byte-Enables, and Alignment 5. Split Transactions 6. Bus Width 7. Source Sampling 8. Compatibility and System Initialization D. Summary of Protocol Rules 1. Basic Initiator Rules 2. Basic Target Rules 3. Bus Arbitration Rules 4. Configuration Transaction Rules 5. **Parity Error** Rules 6. Bus Width Rules E. Registered PCI Command Encoding F. Registered PCI Extended Command Encoding 1. Validated Extended Command 2. Immediate Extended Command G. Registered PCI Attributes H. Byte-Count Transactions 1. Writes 2. Reads I. Byte-Enable Transactions 1. Writes 2. Reads J. Device Select Timing 1. Writes 2. Reads K. Wait States 1. Writes 2. Reads L. Configuration Transactions M. Delayed Transactions N. Split Transactions 1. Basic Split Transaction Requirements 2. Requirements for Accepting Split Completions 3. Split Completion Exception Message 4. Unexpected Split Completion Exceptions O. Transaction Termination 1. Disconnect With Data a. Initiator Termination and Disconnection b. Target Disconnection 2. Target Retry **Termination a. Byte-Count Transactions** b. Byte-Enable Transactions 3. Split Response Termination 4. Master-Abort **Termination a. Byte-Count Transactions** b. Byte-Enable Transactions 5. Target-Abort **Termination a. Byte-Count Transactions** b. Byte-Enable Transactions P. Bus Width 1. Data Transfer Width 2. Address Width Q. Transaction Ordering and Deadlock-Avoidance 1. Ordering and Passing Rules 2. Required Acceptance Rules R. Transaction Sequence Combining and Re-ordering

Drawing Description Text - DRTX (41):
FIG. 39 is a schematic timing diagram of a Master-Abort **Termination of a Byte-Count Transaction** according to the present invention;

Drawing Description Text - DRTX (42):
FIG. 40 is a schematic timing diagram of a Master-Abort **Termination of a Byte-Enable Transaction** according to the present invention;

Drawing Description Text - DRTX (43):
FIG. 41 is a schematic timing diagram of a Target-Abort **Termination of a Byte-Count Transaction** according to the present invention;

Drawing Description Text - DRTX (44):
FIG. 42 is a schematic timing diagram of a Target-Abort **Termination of a Byte-Enable Transaction** according to the present invention;

Detailed Description Text - DETX (13):
Each Registered PCI transaction may include, for example but not limitation, a 4-bit extended command and a 64-bit attribute that carry additional information about the transaction. This information includes: The byte count of an entire initiator operation. An initiator operation may span multiple bus transactions. The initial transaction of each operation is marked so the target knows when to flush buffers containing stale data. Target devices of read operations can use this information to optimize prefetch and buffer management algorithms. Initiator and Operation ID. Each initiator identifies its bus operations (transactions) so complex target devices (such as host bridges) can utilize their buffer management algorithms to optimize the service to individual streams of operations from each master. Split Transaction information. If the initiator requests a long read from memory and the target is capable of splitting the transaction, the target will **terminate the transaction** in a special way, fetch the requested read data, and initiate its own Split Completion transaction to transfer the data back to the original initiator. Split transactions are optional on all other transactions except memory writes, which are posted.

Detailed Description Text - DETX (52):
Protocol rules, according to the present invention, may be divided into the following categories: basic initiator rules, basic target rules, bus arbitration rules, configuration transaction rules, **parity error** rules and bus width rules. The following summarizes the protocol rules for Registered PCI transactions. A more detailed description of these rules will be made hereinbelow.

Detailed Description Text - DETX (54):
The following rules control the way a device initiates a transaction. 1. As in conventional PCI protocol, an initiator begins a transaction by asserting FRAME#. The first clock in which FRAME# is asserted is the address and command phase. (See Section I. (P)(2), Address Width hereinbelow for dual address cycles.) 2. There are no address alignment requirements for beginning a Registered PCI transaction (both byte-count and byte-enable transactions). 3. The attribute phase clock follows the address phase. The attributes include information useful for data-buffer management by the target. 4. Initiator wait states are not permitted. The initiator shall assert IRDY# one clock after the attribute phase. The initiator shall not deassert IRDY# until the end of the transaction or at an ADB boundary. Therefore, data stepping is not possible on write transactions. 5. For write transactions, the initiator shall assert write data on the AD bus no later than 1 clock after it asserts IRDY#. 6. If the transaction is a byte-enable transaction (that is, if it uses an extended command that is designated for byte-enable transactions), the initiator intends to transfer data in only a single data phase. During the address phase the full AD bus indicates the starting byte address of the transaction. The initiator deasserts FRAME# when it asserts IRDY# and uses the byte enables to indicate which bytes of the AD bus are affected by the transaction. (The Operation Byte Count field is reserved.) Byte enables shall be asserted 1 clock after the attribute phase for both reads and writes. Byte-enable transactions are limited to 32-bit data phases. 7. If the transaction is a byte-count transaction (that is, if it uses an

extended command that is designated for byte-count transactions), the following rules apply: a. The transaction address a prefetchable memory location. b. The AD bus specifies the starting byte address of the transaction (including AD[2:0]).
c. The byte count for the operation is included in the attribute field. d. Byte enables are reserved and deasserted (high logic voltage) throughout the transaction. All bytes are included in the transaction from the starting address through the byte count. e. The initiator keeps FRAME# asserted after it asserts IRDY#. f. The initiator is limited to **terminating the transaction** only on naturally aligned 128-byte boundaries called Allowable Disconnect Boundaries (unless the byte count is satisfied sooner). g. A transaction will have less than 4 data phases in the following two cases: 1) The byte count is satisfied in less than 4 data phases, or 2) The transaction starts less than 4 data phases from an ADB and the Disconnect on ADB attribute bit is set. In both of these cases the initiator shall deassert FRAME# two clock after TRDY# asserts for transactions with 3 data phases. If the transaction has 2 or 1 data phases the initiator shall deassert FRAME# with the IRDY# deassertion. If the transaction has 3 or 2 data phases, the initiator shall deassert IRDY# one clock after the last data phase. If the transaction has 1 data phase, the initiator shall deassert IRDY# 2 clocks after the data phase. h. If the byte-count transaction requires 4 or more data phases, the initiator shall terminate the transfer when the byte count is satisfied. The initiator is also permitted to **terminate the transaction** on any ADB. To terminate the transfer, the initiator deasserts FRAME# 1 clock before the last data phase (the last data phase is the clock in which the byte count is satisfied, or the last data phase before crossing the ADB), and deasserts IRDY# 1 clock after the last data phase. i. If the target asserts STOP# 4 clocks before an ADB, the initiator will deassert FRAME# 2 clocks later, and **terminate the transaction** on the ADB.
j. If the transaction is a write and the target inserts wait states, the initiator shall toggle between its first and second data transfers

until the target asserts TRDY#. 8. A Registered PCI initiator is required to repeat all transactions terminated with Retry. There is no exception (as in conventional PCI) for the device being reset. The device driver is not permitted to reset the device, if the device has issued a request that was terminated with Retry or Split Response and that request has not completed (entire byte count transferred). 9. Like conventional PCI, no device is permitted to drive and receive a bus signal at the same time.

Detailed Description Text - DETX (61):
5. **Parity Error** Rules

Detailed Description Text - DETX (62):
The following protocol rules apply to exception conditions. 1. If a device receiving data detects a data **parity error,** it shall assert PERR# on the second clock after PAR is asserted (1 clock later than conventional PCI). 2. During read transactions the target drives PAR on clock N+1 for the read-data it drove on clock N and the byte enables driven by the initiator on clock N-1. 3. All Registered PCI device adapters are required to service PERR# conditions for their transactions. See the section titled "Error Handling and Fault Tolerance." 4. Whether a device decodes it address during the address phase or not, if that device detects a **parity error** on an attribute phase, the device asserts SERR#. Other SERR# and status bit requirements for address-phase and data-phase errors are the same as for conventional PCI.

Detailed Description Text - DETX (77):
A Registered PCI target that detects a reserved validated extended command shall **terminate the transaction** with Target-Abort. Table 8 lists the validated extended commands contemplated for the present invention.

Detailed Description Text - DETX (93):
If the target asserts TRDY# on a byte-enable transaction, the target does not assert STOP#. See Section I. (O)(2)(b), Byte-Enable Transactions associated with FIG. 37 and Section I. (O) (5)(b), Byte-Enable Transactions associated with FIG. 42 hereinbelow for the use of STOP# in Retry and Target-Abort **termination of byte-enable transaction**.

Detailed Description Text - DETX (109):
Wait states should not be used, but if necessary, the number of wait states
shall be kept to a minimum. Preferably, **terminating the transaction** with Retry
or executing it as a Split Transaction will provide more efficient use of the
bus. In many cases the transaction also completes sooner. The use of Retry
and Split Transactions is preferred unless there is a high probability that
inserting target wait states will be faster. Even in high-frequency systems
with few (maybe only one) slots, Retry **termination and Split Transactions** allow
multi-threaded devices (e.g. multiple devices behind a PCI-to-PCI bridge) to
issue multiple transactions concurrently.

Detailed Description Text - DETX (120):
For the initiator: 1. The initiator is shall repeat the full starting
address specified on the AD bus down to the byte, including AD[2:0] for all
transaction terminated with Retry. 2. The initiator shall repeat all
attributes for all transaction terminated with Retry 3. The initiator shall
repeat a transaction **terminated with Retry, until the transaction** completes.
Device drivers are not permitted to reset any device with an outstanding
transaction (either Delayed Transaction or Split Transaction)

Detailed Description Text - DETX (124):
Only memory-read transactions that use byte-count protocol use Split
Transactions. The target of any byte-count memory-read transaction may
optionally complete the transaction as a Split Transaction or may use any other
termination method (immediate response, Retry, or Delayed Transaction), as
determined by the rules for those termination methods. All of these
termination alternatives are available regardless of whether the transaction
was previously **terminated with Retry, or whether the transaction** is the initial
transaction of a Sequence (i.e. the Initial Sequence Request attribute bit is
set) or a continuation of a Sequence previously disconnected after transferring
some data. Once the target terminates a byte-count memory-read transaction
with Split Response, the target shall transfer the entire remaining byte count
as a Split Completion (except for error conditions described

hereinbelow).

Detailed Description Text - DETX (125):
A Split Transaction begins when an initiator (called the requester) initiates a memory-read transaction using a byte-count extended command. If a target (called the completer) that supports Split Transactions is addressed by such a transaction, it may optionally signal a Split Response termination by doing the following: 1. Assert DEVSEL# in the response phase to claim the transaction on clock N. 2. Assert STOP# on clock N+1. 3. Assert TRDY# on clock N+2. 4. Deassert TRDY#, **STOP#, and DEVSEL# to terminate the transaction** on clock N+3.

Detailed Description Text - DETX (136):
To prevent one requester from consuming more system buffer space than is appropriate, the length of time a requester can hold off accepting Split Completion transactions is limited. Preferably, the requester shall not hold off a Split Completion transaction for more than 1 .mu.s in any 20 .mu.s period. This preferred Split Completion hold-off time is calculated as follows for any 20 .mu.s period: ##EQU1## H=total hold-off time h.sub.n =n.sub.th hold-off time for a Split Completion. h.sub.n is measured from the time the requester asserts **STOP# to disconnect or terminate the transaction** with Retry, until the requester asserts TRDY# for a subsequent Split Completion transaction. N=number of Split Completions in the 20 .mu.s period, for all Split Requests from a single requester.

Detailed Description Text - DETX (145):
Abnormal conditions can also occur in the second phase of a Split Transaction, after the completer has terminated a byte-count memory-read transaction with Split Response termination. Such conditions can prevent the completer from executing the request. Examples of such conditions include the following: 1. **parity errors** internal to the completer. 2. the byte count of the request exceeding the range of the completer.

Detailed Description Text - DETX (151):
If the requester detects a data **parity error** during a Split

Completion, it
asserts PERR# if enabled and sets bit 15 (Detected **Parity Error**) in the Status
register. The requester also sets bit 8 (Master Data **Parity Error**) in the
Status register, because it was the original initiator of the Sequence (even
though the completer is the initiator of the Split Completion).

Detailed Description Text - DETX (160):
Transactions that are disconnected by the initiator on an ADB before the
byte count has been satisfied and those that terminate at the end of the byte
count appear the same on the bus. The following figures illustrate initiator
disconnection or **termination for transactions** with 4 or more data phases and
for transactions with less than four data phases. In FIG. 28, the initiator
disconnects or terminates after 4 or more data phases. In this case the
initiator signals the end of the transaction by deasserting FRAME# 1 clock
before the last data phase. In FIGS. 29-31, the initiator disconnects or
terminates after 3, 2, and 1 data phases, respectively. In this case the
initiator signals disconnection by setting the Disconnect on First ADB
attribute bit. (Termination occurs at the end of the byte count). In FIG. 29,
the initiator deasserts FRAME# for a 3 data phase transaction, 1 clock after
the Initiator detects the assertion of the sampled TRDY#. In FIGS. 30 and 31,
the initiator deasserts FRAME# for a 2 and a 1 data phase transaction,
respectively, showing the FRAME# deassertion occurring with IRDY#.

Detailed Description Text - DETX (167):
Referring to FIG. 36, a schematic timing diagram of a Byte-Count Transaction
Showing Target Retry **termination of a byte-count transaction** is illustrated.

Detailed Description Text - DETX (169):
Referring to FIG. 37, a schematic timing diagram of a Byte-Enable
Transaction Showing Target Retry **termination of a byte-enable transaction** is
illustrated.

Detailed Description Text - DETX (174):
Referring to FIG. 39, a schematic timing diagram of a Master-Abort
Termination of a Byte-Count Transaction is illustrated.

Detailed Description Text - DETX (176):
Referring to FIG. 40, a schematic timing diagram of a Master-Abort **Termination of a Byte-Enable Transaction** is illustrated.

Detailed Description Text - DETX (180):
FIG. 41 illustrates a schematic timing diagram of a Target-Abort **termination of a byte-count transaction**.

Detailed Description Text - DETX (182):
FIG. 42 illustrates a schematic timing diagram of a Target-Abort **termination of a byte-enable transaction**.

Detailed Description Text - DETX (205):
The Registered PCI invention has the same requirement for accepting transactions as conventional PCI, with one exception described hereinbelow. Using the terminology of the PCI Specification incorporated by reference herein, a "simple device" (i.e. one that does not do outbound write posting) can never (with the exception described hereinbelow) make the acceptance of a transaction as a target contingent upon the prior completion of another transaction as an initiator. A "bridge device" (i.e. one that does outbound write posting) can never make the acceptance (posting) of a memory-write transaction as a target contingent on the prior completion of a transaction as an initiator on the same bus. Furthermore, to provide backward compatibility with PCI-to-PCI bridges designed to revision 1.0 of the PCI-to-PCI Bridge Architecture Specification, incorporated by reference herein, all devices are required to accept memory-write transactions even while executing a previously enqueued Delayed Transaction. A device is permitted to **terminate a memory-write transaction** with Retry only for temporary conditions that are guaranteed to resolve over time (as limited by the Max Completion ECR). Bridge devices are permitted to refuse to accept non-posted transactions as a target until the device completes its memory-write transactions as an initiator.

Detailed Description Text - DETX (254):
Although it is allowable for a bridge device to assert SERR# on the **detection of a error** condition (usually as the first point of failure), this

action is not encouraged for bridge designs that are optimized for error
recovery. A bridge design that is optimized for error recovery is also designed to forward bad parity (just like conventional PCI), and will not
interrupt any transactions as a result of a data parity exception. This
results in transaction running to completion on Byte-Count even if a PERR# is
asserted. When this condition occurs, the bridge must forward the exact data
including the PAR and PERR# status for the data. In addition, the bridge must
update its PCI Status register to reflect its detection of the exception.
However, no further action must be taken by the bridge, which allows the error
condition to be handled by the original initiator of the transaction (i.e., the
originating device).

Detailed Description Text - DETX (263):
If a device receiving data detects a data **parity error,** it must assert PERR#
on the second clock after PAR is asserted as illustrated in FIGS. 50 and 51.
Note that this is one clock later than conventional PCI. All Registered PCI
device adapters are required to service PERR# conditions for their transactions. See the section titled "Error Handling and Fault Tolerance."

Detailed Description Text - DETX (264):
Whether a device decodes it address during the address phase or not, if that
device detects a **parity error** on an attribute phase, the device asserts SERR#,
if enabled. Other SERR# and status bit requirements for address-phase and
data-phase errors are the same as for conventional PCI.

Detailed Description Text - DETX (268):
Registered PCI error handling builds on conventional PC error functions in
order to provide a more fault-tolerant system. All Registered PCI devices and
their software device drivers are required either to recover from the data
parity error or to assert SERR#. In conventional PCI systems, error-recovery
mechanisms are only suggested. This optional error recovery support forces
most systems to handle a data **parity error** condition as a catastrophic error
that will bring the system down. Usually, the only service that is

done by the
system is to log the error, notify the user of the error condition, and execute
a system halt instruction from the CPU to shut down the system. By requiring
the device and device driver either to recover from the error or to assert
SERR#, the system is freed from the assumption that all errors are catastrophic
errors.

Detailed Description Text - DETX (269):

If a data **parity error** (calculated data **parity error** on a read or PERR#
asserted on a write or Split Completion), is detected by a Registered PCI
initiator, the Registered PCI initiator is required to provide one of the
following levels of support for data **parity error** recovery: 1. Assert SERR#.
Notice that no device driver support is required for this technique.

Detailed Description Text - DETX (287):

The electrical interface requires universal I/O buffers (as defined in the
PCI Local Bus Specification) for all Registered PCI devices that support
operation in 5 V I/O, 33 MHz buses. Registered PCI devices may optionally
operate only in 3.3 V **I/O buses**.

Detailed Description Paragraph Table - DETL (5):

TABLE 5 Registered PCI DEVSEL# Timing Decode Speed Registered PCI
Conventional PCI 1 clock after address Not Supported Fast 2 clocks after
address Decode A Medium 3 clocks after address Decode B Slow 4 clocks after
address Decode C SUB 6 clocks after address SUB N/A 2. The target asserts
TRDY# an odd number of clocks after it asserts DEVSEL#. The target cannot
deassert TRDY# until the end of the transaction. Target wait states are not
permitted after the initial data phase. Therefore, data stepping is not
possible after the first data phase of read transactions. 3. A Registered PCI
target is required to meet the same target initial latency requirements as a
conventional target. That is, the target shall assert TRDY# or STOP# within 16
clocks from the assertion of FRAME#. Host bridges are allowed to extended this
time in some cases. See Section I. (K), Wait States hereinbelow for details.

4. If the transaction uses a byte-enable extended command, the target shall assert TRDY# an odd number of clocks after it asserts DEVSEL#, and deassert TRDY# and DEVSEL# together one clock later. 5. Targets shall alias reserved command encoding 1100b to Memory Read and 1111b to Memory Write. 6. If the transaction uses a byte-count extended command, the following rules apply: a. The target is limited to **terminating the transaction** only on naturally aligned 128-byte boundaries called Allowable Disconnect Boundaries (ADB) (unless the byte count is satisfied sooner). b. If the byte count is satisfied, the target deasserts TRDY# on the clock after the last data phase of the transaction. c. To **terminate the transaction** on an ADB after 4 or more data phases the target asserts STOP# 1 clock before the last data phase (i.e. 2 clocks before crossing the ADB), and deasserts TRDY# and STOP# 1 clock after the last data phase. d. If the transaction starts 1, 2, or 3 data phases from an ADB, and the target wants to disconnect the transaction on the ADB, the target asserts STOP# when it asserts TRDY#. The target then deasserts STOP# when it deasserts TRDY#. 7. If the transaction is one of the special termination cases that do not transfer data, the target shall assert and deassert DEVSEL#, TRDY# and STOP# according to the following cases: a. Target-Abort on the first data phase--The target deasserts DEVSEL# and asserts STOP# together an odd number of clocks after it asserts DEVSEL#. The target deasserts STOP# 1 clock later. TRDY# is not asserted. b. Retry termination--The target asserts STOP# an odd number of clocks after it asserts DEVSEL#. The target deasserts STOP# and DEVSEL# 1 clock later. TRDY# is not asserted. c. Split Response termination--The target asserts STOP# an odd number of clocks after it asserts DEVSEL#, and asserts TRDY# one clock later with all bits on the AD bus driven high. The target deasserts STOP#, DEVSEL#, and TRDY# together 1 clock after it asserts TRDY#. 8. Like conventional PCI, no device is permitted to drive and receive a bus signal at the same time. 9. Prefetchable memory address ranges for all devices are no smaller than 128 bytes. If an initiator issues a byte-count memory-read to a device starting at an implemented address and

proceeding to the next ADB, the device shall return FFh for each unimplemented byte.

US-PAT-NO: 5517626

DOCUMENT-IDENTIFIER: US 5517626 A

TITLE: Open high speed bus for microcomputer system

---------- KWIC ---------

Brief Summary Text - BSTX (5):
I/O (input/output) bottlenecks are typically present in current PC (personal computer) architectures. In the IBM PC AT/MCA or industry standard EISA architecture, the CPU (central processing unit) is closely coupled with I/O peripherals. That coupling occurs through the **I/O bus,** which is simply a buffered CPU local bus. With this tightly coupled architecture, any bus activity by peripherals on the **I/O bus** will affect the CPU and system memory performance. As CPU speed increases, the I/O performance penalty is multiplied.

Brief Summary Text - BSTX (15):
The status signal line includes a line for carrying a signal for indicating the slave having provided valid data in response to a read or having accepted data in response to a write; a line for carrying a signal for indicating the slave can fulfill the request of the master for transfers; a line for carrying a signal for indicating the data transfer being able to be burst transferred to or from the slave; a line for carrying a signal for indicating the slave being not ready to handle the requested transaction to cause the requesting master to back off the bus; a line for carrying signal asserted by any participating slave for extending the snoop cycle of the current transaction; a line for carrying a signal asserted by any participating slave for indicating a **parity error** on the data lines during a data cycle; a line for carrying a signal for indicating that a non-master cache controller will retain a copy of addressed data in a current transaction; a line for carrying a signal for indicating whether the current transaction is cacheable; and a line for carrying a signal

for indicating a processor controller intervening in the transaction.

Detailed Description Text - DETX (94):
Any participating slave can assert SBPARITY# if it detects **parity error** on SD[31:0] during any of the data cycles. The SA[31:0] and the SD[31:0] that caused the parity problem should be recorded in the CSR of the parties that caused the error (in this case, both the active master and the selected slave.)

Detailed Description Text - DETX (96):
System error is flagged anytime a ACI master or participating slave **detects an error** that is fatal or requires attention. Usually the master-slave pair that caused the system errors would have the violating conditions stored in the corresponding CSRs to assist later probing with software via traps or other vehicles. ACI masters or slaves are not affected by SYERR# status.

Detailed Description Text - DETX (200):
3.7.4.1. **Interrupt Acknowledge Transactions**

Detailed Description Text - DETX (257):
The backoff mechanism is implemented using the SBOFF# line. It is provided in order to avoid deadlock on the bus. A slave can **interrupt an ongoing transaction** or it can prevent further transactions from being initiated. The use of backoff in this architecture is to prevent deadlocks between the ACI and the **I/O bus**. A backoff is initiated when a Master wants to perform an operation on the **I/O bus** (ISA, EISA or MCA) and at the same time the I/O Controller (ISA, EISA or MCA) wants to initiate a transfer on the ACI. If the SBOFF# line is asserted anytime within an on-going transaction, the current bus master withdraws from the bus during the next bus clock cycle and tristates all of its signals. The bus arbitration will continue to run, but all bus grants will be driven inactive and the MCA or EISA controller will be given the ACI.

Detailed Description Text - DETX (264):
1) CD.sub.-- ChCK (channel memory) 2) Channel timeout (DMA timeout) 3) Watch dog timeout (INT 0) 4) ACI memory **parity error**

Detailed Description Text - DETX (373):
C--Those residing on the **I/O bus** and providing their own CSR data and buffer
enables (e.g. Peripheral Interface Controller). These devices have IDs in the
range 4000h to 7FFFh. Writes to 22, 23, 25 and 27 Propagate to the **I/O Bus**.
Reads from 27 Propagate to the I/O only for devices in classes B and C.

Detailed Description Text - DETX (608):
bit 0 BPC, bus parity control when set to 1, parity is checked when data is
received from the bus, and parity is passed to the bus if there is no **parity error,** or parity is generated for ECC memory, when clear bus parity is ignored.

Detailed Description Text - DETX (625):
Any time when memory read/write problems occur with the data, either **parity error** in the case of parity memory, or two-bit **detect errors** or 1-bit correct
situation will cause SYERR# and SBPARITY# to be asserted in the bus. Four
consecutive Status Registers are used to remember the pre-mapped address of the
memory location that caused the error.

Detailed Description Text - DETX (669):
The Memory Controller asserts SBPARITY# if it detects **parity error** on
SD[31:0]during any of the data cycles. The SA[31:0] and the SD[31:0] that
caused the parity problem should be recorded in the CSR of the parties that
caused the error in this case, both the active master and the Memory Controller.

Detailed Description Text - DETX (1179):
This bit when set indicates that a NMI has occurred and forces all
arbitrations to be terminated. NMI could have occurred because of a **parity error,** channel check error, watch dog timeout error or a bus timeout condition.
On a power on reset, this bit is set and allows the CPU to run system wide
diagnostics (typically POST) and configurations. When this is done the CPU
clears this bit enabling the system for arbitrations. Once this bit is cleared
by the CPU, only a NMI should be allowed to set his bit i.e. an application

should not set this bit.

Detailed Description Text - DETX (1364):
The system provides 16 levels of system interrupts. Any or all of the interrupts may be masked, including non-maskable interrupt (except for the Watchdog timer). NMI has the highest interrupt priority. It signals the system microprocessor that a **parity error,** a channel check, a system channel time-out, or a system watchdog timer time-out has occurred.

Detailed Description Text - DETX (1423):
The Watchdog timer detects when IRQ0 is active for more than one period of OUT0. If IRQ0 is active when a rising edge of OUT0 occurs, the count is decremented. When the count is decremented to 0, an NMI is generated. Thus, the Watchdog timer may be used to detect when IRQ0 is not being serviced. This is useful to **detect error** conditions.

Detailed Description Text - DETX (1510):
The NMI mask bit resides in bit 7 of port 0070H. Writing a 1 to this bit inhibits interrupts due to a memory **parity error** or a channel check. The mask bit does not inhibit interrupts due to the Watchdog timer or system channel time-out. The least significant 6 bits provide the address for the RTC/CMOS RAM

Detailed Description Text - DETX (1521):
The keyboard/auxiliary device controller is a function of the Intel 8042 chip. The keyboard is connected to one of the two connectors in the rear of the system unit. This connector is dedicated to the keyboard. An auxiliary device connects to the other controller connector. The auxiliary device may be any type of serial input **device compatible with the 8042 interface**. It could be a mouse, touch pad, track-ball or keyboard.

Detailed Description Paragraph Table - DETL (11):
_____________________________________ bit 7 Parity check 1 writing 1
resets IRQ0 (affects PIC only) writing 0 has no effect reading 1 means parity

has occurred indicates that parity has been checked 0 not checked yet

Writing bit 6:3 reserved and ignored bit 2 parity enable (powers up set) 1
disable and clear parity set during power-on reset 0 enables parity check
1-0 transition re-arms parity check in MCA machines, registered **parity error**
is reported immediately in AT machines, registered **parity error** is forgotten
bit 1:0 reserved and ignored

Detailed Description Paragraph Table - DETL (69):
2.1 BUS INTERFACE SIGNALS
SD[31:0] 1-32 **I/O The Bus** data lines.
SDP[3:0] 33-36 I/O Four even parity bits, each indicating the even parity for
each byte on the S3 data Bus. SLSB 37 I/O 486 only command/copy back command. When SLSB is high, only 486-like commands are used. When it is low,
copy back masters are on the bus besides the write-through masters. SM/IO#
38 I/O The memory/input-output line defines the bus cycle as a memory cycle
or an IO cycle. When it is active low, the cycle is an IO cycle. SD/C# 39
I/O The data/control line defines the bus cycle as a data access or code
access. When it is active low, the cycle is a code operation. SW/R# 40 I/O
The write/read line defines the bus cycle as a write cycle or read cycle.
When active low, the cycle is a read cycle. The instruction command encoding
for 80386 SM/IO# SD/C# SW/R# 0 0 0 Interrupt Acknowledge 0 0 1 Reserved 0 1
0 IO Read 0 1 1 IO Write 1 0 0 Code Read 1 0 1 Special Cycle 1 1 0 Memory
Read 1 1 1 Memory Write The instruction command encoding for 80486 is:
SM/IO# SD/C# SW/R# 0 0 0 Interrupt Acknowledge 0 0 1 Special Cycle 0 1 0 IO
Read 0 1 1 IO Write 1 0 0 Code Read 1 0 1 Reserved 1 1 0 Memory Read 1 1 1
Memory Write SLOCK# 41 I/O This indicates the bus cycle is normal cycle or
locked cycle. When active low, the cycle is a locked cycle. SBE[3:0]# 42-45
I/O The byte enable signals indicate which bytes are active during a read or
write. SADS# 46 I/O The address status enable indicates that valid address
and status on the bus. SRESET# 47 I Reset forces the chip to known states.
SRDY# 48 I/O Non-Burst ready indicates the end of the current bus cycle. This

used to signal the master that the slave has provided valid data for read
and accepted data for write. SBRDY# 49 I Burst ready input, it indicates
that the slave has provided valid data for read and accepted data for write.
SBLAST# 50 I/O Burst last cycle, it is driven by the controller when the
master access the local bus. SWAIT# 51 I Local bus wait signal, it causes
the controller to ignore the SRDY# and SBRDY# signals SBOFF# 52 O Back off.
It, forces the CPU to float its bus for the next transaction cycle.
SBREQ#[3:0] 53-56 I Local bus request, For centralized arbitration, one bus
request signal is assigned to each of four masters on local bus.
SBGRANT#[3:0] 57-60 O Local bus grant, one bus grant signal is assigned to
each of four masters on local bus. SCLK 61 I local bus clock SHOLD 62 O Bus
hold request to allow the controller complete control of 486 microprocessor
bus SHLDA 63 I Hold acknowledge goes active in response to a hold request.
SHLDA goes active to indicate the 486 has given the 486 bus to another local
bus master.

Claims Text - CLTX (1):

1. A local system bus for a micro-processor based computer system including a plurality of controllers connected between the local system bus and at least one micro-processor, a system memory or an **I/O bus,** one of the controllers being a bus master and another being a bus slave during a bus transaction, the local system bus comprising:

Claims Text - CLTX (24):

18. A local system bus as in claim 1, wherein the lines for carrying status signals comprise a line for carrying a signal indicating a **parity error** on the lines for carrying data during a data cycle.

Claims Text - CLTX (41):

35. A local system bus as in claim 1, wherein the micro-processor based computer system comprises a peripheral control bus connected to the **I/O bus** and to each controller for programming peripheral devices connected to the local system bus.

Claims Text - CLTX (57):
an **input/output bus** for connecting to the input/output devices;

Claims Text - CLTX (58):
a first controller for connecting the local system bus to the **I/O bus**;

Claims Text - CLTX (59):
wherein the local system bus operates at a higher data rate than does the
input/output bus;

Claims Text - CLTX (62):
40. A computer system as in claim 39, wherein the **input/output bus** is one
of an ISA, MCA or EISA bus.

Claims Text - CLTX (63):
41. A computer system as in claim 39, further comprising a peripheral
control connected to the **input/output bus** and at least one cache controller for
programming peripheral devices connected to the local system bus.